無線LAN内蔵SDカードFlashAirのちょっと変わった使い方を中の人が詳しく解説！トップギアで始めよう！

Contents

FlashAir 応援キャラクター「閃ソラ」

閃 ソラ（ひらめき そら）は、FlashAir の非公式応援キャラクターです。とある航空会社の CA をしています。フライトでたびたび不在にしますが、オフにはミラーレス一眼で写真を撮ったり、電子工作したり、アプリ開発したり、忙しい毎日を送っています。

■ プロフィール

名前	閃ソラ（ひらめき そら）
年齢	23 歳
職業	新人キャビンアテンダント
趣味	電子工作、アプリ開発
悩み	飛行機の中で電子機器が使えないこと
Twitter	@Hirameki_Sora

Maker Faire Tokyo 出展にあたり

高田

FlashAir をどんな用途で使えば良いのか、何度もディスカッションや検討を繰り返して痛感したのは、製品をつくる側だけでは用途開発はできない、ということでした。キラーアプリを探そう、なければどこかと一緒につくろう、と模索を続けていますが、企業同士での活動は早い段階で事業性の目処がたったものでないと推進力がつきません。アイデアディスカッションをしても、ディスカッションの結果をまとめて終わってしまいます。もっと違う形で、キラーアプリにつながる活動はないものか？アイデアを出してくれる人たちと活動できないだろうか？と考えていたときに紹介してもらったのがオープンソースカンファレンスの活動でした。ここに集う人たちのものづくりにかける情熱！（それなのに熱さが表に出ない独特のパフォーマンス）。プロダクトに関する素直な期待とコメント。そこには大企業同士の案件を中心にまわる事業開発とはまったく別の世界が待ち受けていました。それは、エンジニア個人を基本にした純粋な興味、関心を原動力に技術そのものを探求する活動だからこそでてくるパワーなのだと感じます。

そんな世界に FlashAir もおずおずと足を踏み出してみました。そもそも素材として面白い、と思ってもらわなければ始まりません。皆様に興味を持っていただけるよう、Maker Faire Tokyo 2014 には FlashAir を使った電子工作の作品を多く集めました。これほどの数が集まるとは予想していませんでしたので、小さいブースに展示品がてんこ盛りですが、是非とも見ていただければと思います。そして、興味を持っていただけたら、FlashAir で遊んでみてください。作品ができたら公開してください。

この活動をはじめてみて、FlashAir はひとつのキラーアプリで世の中に残る商品ではないと感じてます。仕事や日常生活のちょっとした不満や「あったらいいな」、を簡単に実現できるツールとして、多様な使い方で身近に存在するもの、それが FlashAir がこの何年か先に見せる姿になると思えて仕方ありません。

生活に役立つものはもちろんのこと、ゲームやおもちゃ、ほっとリラックスできるもの、あほらしくて笑っちゃうのにくせになるもの、学びを助けるもの、そんな多様な活用アイデアがこの活動から出てくることを願っています。

遊べるなぁ FlashAir、と言ってもらえるように。そして、その先に FlashAir を使った日本発の IoT を世界に紹介できるといいなぁ、という野望を秘めつつ、息の長い活動にしていきたいと考えています。

FlashAir と GPIO

伊藤

我が家には３歳の息子がいる。名前はＫくんだ。一般的に男の子は女の子に比べて言葉を覚えるのが遅い。だから最近やっと会話っぽいのができるようになってきた。そんなＫくんはお化けが怖いらしく、お化けの話をすると「やめて！」と叫んで泣きそうになる。土日はそんな子と遊んでいるのが私としてはとても楽しい。

そのＫくんが生まれた時の話だけれど、生まれてからちょうど２週間後に FlashAir の原型となるカードの形をしたサンプルが出来上がった。まあこちらもある意味生まれたての状態ではあるが、まだ何もできなかった。

今でこそ、FlashAir というのは SD カードに無線の AP と Web サーバーを入れたもので、「カメラで撮った写真をその場でスマホに転送する」というコンセプトになっているが、当時は全くそんな話も無く、実は中で動くソフトさえも無い状態だった。なので、サンプルができた後にＫくんを膝の上において、せっせとノートＰＣでコーディングしていたのは、今では飲み会のネタである。

そんなＫくんが、最近お願いするとちょっとしたことをやってくれるようになった。
「電気のスイッチつけて。」
「寝るから絵本とって」
とか、そんな簡単なことである。
でも、親としてはなんだか自分の子とコミュニケーションがとれているようでちょっと嬉しくなってくる。この子は小さいながらも言葉を理解して成長しているんだなという事を実感する。
そんな３歳の子供にお願いできるようなとても簡単な事を、もう一つの子供のようなものでもある FlashAir でもできるようにしてみた。それが FlashAir に GPIO をつけてみたよという話。

ここで、GPIO というのは General Purpose Input Output の略である。どういう事かというと、例えばマイコンの端子から出てくる電気信号を1とか0とかにすることができますよ、という話。そのマイコンの GPIO をつなげてやれば例えば扇風機のスイッチを入れる事ができるかもしれないし、LED ライトをつける事もできたりするかもしれない。

そんな些細な事を実現する事ができる GPIO だが、FlashAir でどうやって GPIO を使うかと言うと、３歳の子供にお願いするのと一緒である。
例えば、
・Ｋくんに声でお願いする代わりに、FlashAir に無線 LAN を使ってお願いする。
・Ｋくんがスイッチを入れる代わりに、FlashAir の SD 端子を1とか0とかにする。
・「お願い」って言葉の代わりに、ブラウザで http://~ って言ってみる。

そんな FlashAir の GPIO 機能だけど、Ｋくんと違って、FlashAir は素直に言うことを聞いてくれます。けれども、片手ぐらいの５本くらいしか1とか0にすることができないのでやれる事は同じくらいかな。まあ、たまに通信できなくなるのも一緒かもしれないけどね。詳しいことは他の人が説明してくれそうなので割愛します。ただ、みんなもそんな感じで FlashAir で遊んでくれて Twitter でつぶやいてくれたらとっても嬉しいですね。

※この話はフィクションです。

OSC で、ソラちゃんと、GPIO ができるまで

Pochio

今でこそ展示会のブースに立つと FlashAir を知っているという多くのお客様とお会いできますが、1 年前は「知っている」と言ってくださるお客様に出会うこと自体がまれでした。それもそのはず、実は私もそのちょっと前まで、自社製品でありながら FlashAir の存在を全く知らなかったのです。

FlashAir との出会い

FlashAir との出会いは突然でした。2013 年の 6 月中旬、元部長が FlashAir のパッケージを持って私の隣の同僚に、「面白い使い道はないか?」と相談に来ていました。元部長が去った後、「こんなのもらった」とその同僚から見せてもらったのが、FlashAir の青いパッケージとの初めての出会いでした。

FlashAir はデジタルカメラでの利用を念頭に置いた製品ですが、無線 LAN がついた SD カードと聞いてすぐに頭に浮かんだのが、Raspberry Pi（ラズベリーパイ）でした。Raspberry Pi とは世界的に流行っている Linux が動く小型のコンピュータです。インターフェースとして GPIO（汎用入出力）端子と 2 つの USB ポート、有線 LAN ポートと SD カードスロットが搭載されています。無線 LAN を使いたい場合は、USB ポートに無線 LAN モジュールを挿さなければなりません。しかし、もし SD カードスロットに挿した FlashAir から無線 LAN 通信ができれば、USB ポートを塞がなくて済むため、世界中の Raspberry Pi ユーザーがきっと使ってくれるのではないか、と考えたのです。

急いで FlashAir と Raspberry Pi との組み合わせについての説明資料を作って FlashAir の関係者に送ったところ、本社での打ち合わせに呼び出されました。しかしここで FlashAir を無線 LAN カードのように使用できないことが判明し、目論見はあっさり外れてしまったのです。それでも Raspberry Pi との組み合わせに可能性を感じていたので、そのユーザーである私の旧知の Dr. K 氏にご相談しました。

頂いた貴重なご意見が、

(1)FlashAir に GPIO をつけるべき

(2)FlashAir 上でプログラムが動くようにするべき

(3)FlashAir の API を公開しているので、OSC[1] に参加して宣伝してはどうか？

というものでした。(1) と (2) は組み込み用途で使うために必要な機能というご意見でしたが、簡単には実現できません。まず (1) については「カードの端に穴を開けて、そ

1　オープンソースカンファレンスの略。オープンソースに関するセミナーと展示を主とするイベントで、全国各地で開催されている。

こから端子を出せば」とのリクエストでしたが、FlashAir に穴を開けると SD カード規格から外れてしまいます。規格を作った会社自らそんなことはできません。(2) についてはFlashAir に搭載されている RAM が小さいため、ユーザーの作ったプログラムを実行させることができません。FlashAir には部品がぎっしりと詰まっており、これ以上大きなRAM を搭載する余地がないのです。実現できそうなものは (3) しかありませんから、早速 OSC を運営する「びぎねっと」さんに Dr. K 氏と関係者とでお伺いし、10 月に開催される OSC 東京 2013 秋にて初出展することになりました。

OSC 初出展

出展が決まってから、展示内容について試行錯誤が続きました。ノベルティは当初FlashAir 型のノートを作ろうとしたのですが、SD カードの形になるように右上角をカットしてくれる業者が見つからず、代替案であった付箋紙を作ることになりました（図 3-1）。ブースでは FlashAir の機能を説明し、別室で FlashAir デベロッパーズ[2] のセミナーを開催、来場者には抽選でオリジナルデザインの FlashAir（図 3-2）をプレゼントしました。ところがほとんどのお客様が FlashAir をご存じなく、「Wi-Fi がついた SD カードです」と説明すると、同じく Wi-Fi 機能が搭載された「あのオレンジ色の SD カードと何が違うのか」と聞かれる始末。あのカードとの知名度の差を痛感したのでした。

東京で開催される OSC は年２回あります。10 月の秋の OSC に出展したあと、次の春の OSC（翌年の２月末）にも出展することになりました。実は私の本業は FlashAir とは関係がないため、このあたりで手を引こうと考えていたのですが、FlashAir のスタッフが様々なことに挑戦するのが面白くて、怒られるまでお付き合いさせて頂こうと考えるようになりました。

図 3-1: FlashAir 付箋紙
（上 : 表側、下 : 内側）

図 3-2: オリジナルデザイン
FlashAir 第 1 弾

2　https://flashair-developers.com

年が明けて1月になり、webのニュースをみて衝撃が走りました。半導体界の巨人と呼ばれるあのI社さんから、Edisonが発表されたのです。当初はSDカード型のコンピュータで、SoC、メモリ、ストレージ2GBとWi-Fiを搭載し、カードの裏面の穴からGPIO端子がのぞいていました。そう、まさしくかつてDr. K氏に指摘されたアレです。twitterで早速「それ見たことか」と突っ込まれて心配になり、FlashAirを開発している伊藤さんにこの件を伝えたのですが、意外にも「消費電力高そうですね〜」と意に介さない様子でした。

ソラちゃんの誕生

図3-3: ソラちゃん初期デザイン

図3-4: ソラちゃん決定デザイン

2014年の春のOSC東京は、記念すべきOSC開催100回記念として、協賛各社のマスコットキャラクターのコスプレ大会をすることになりました。その様子はニコ生で中継されることになり、OSCでも人気の「このはちゃん」や「クラウディアさん」などの参加で、盛り上がるのは間違いありませんでした。このイベントへの参加のお誘いを頂いたのですが、正直悩みました。言うまでもなくFlashAirには萌えキャラがいません（涙）。しかしプロモーションの機会として、こんなチャンスはそうそうありません。最近では様々な業種で各社とも萌えキャラを活用した広告を展開しています。ここは「とにかくやってみる」の精神で、FlashAirの萌えキャラを作ってコスプレ大会に乗っかりましょう！と提案したところ、OKを頂きました。

早速関係者とキャラクターの設定を検討し、「23歳のキャビンアテンダント（CA）で、電子工作好きで、悩みはフライト中に機内で電子機器が使えないこと」をコンセプトにしました。ちなみにCAになった理由は、FlashAirの最初のプロモーションで使ったCAの衣装が1着だけ残っていて、コスプレするにもこの衣装しか用意できなかったからです。名前はキャラクターをデザインしたじむさんが「閃(ひらめき) ソラ」と名づけました。Flash(閃光)とAir（空）からつけたベタなネーミングです。じむさんが描いた最初のソラちゃんのデザイン（図3-3）は、のちに衣装に合わせて図3-4のように変更されました。また、じむさんはソラちゃんにオリジナリティを持たせるため、髪の毛が羽でできているという設定を加えました（図3-5）。

1 月末にコスプレのモデルさんの選択会議があり、10 人ほどの候補者の中から「一番かわいい」との声があった國井亜里沙さん[3] にお願いしたいと申し出たところ、即決定となりました。モデルさんが決まり、衣装は既にありますから、残りはソラちゃんの髪の毛が羽でできているという無茶な設定の実現です。これをどうにかできればコスプレ界での Leading Innovation です。じむさんがウィッグを製作している美容室に電話しまくり、秋葉原にある美容室が対応してくださることになりました。

そして 3 月 1 日の本番を迎えました。この日は朝 7 時過ぎに OSC の会場である明星大学に行くと、すでに國井さんがお越しで、じむさんも完成したウィッグ（図 3-6）を持参していました。プロのメイクさんにより完成品のウィッグを装着した國井さんは、それはもうソラちゃんそのもので、見事と言うほかありませんでした。今でも Google で「閃ソラ」で画像を検索して頂くと、その様子を見ることができます。セミナーでは前回同様、図 3-7 のデザインの FlashAir をプレゼントしました。秋と春の 2 回の OSC 参加で、ギーク（技術オタク）な方々に対して FlashAir の知名度をかなり向上させられたのではないか、と思っています。

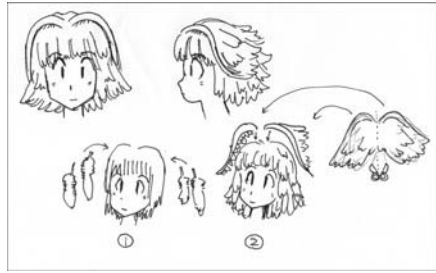



図 3-5: ソラちゃんの髪の毛の設定

図 3-6: ソラちゃんのウィッグ (試作品)

図 3-7: オリジナルデザイン FlashAir 第 2 弾

FlashAir Make Idea コンテストの開催と GPIO の搭載

6 月になると、FlashAir デベロッパーズのサイトにて FlashAir の活用アイディアを募集する「FlashAir Make Idea コンテスト」を開催することになりました。このイベント告知のチラシを厚手の紙の「うちわ」で作成し、7 月の OSC 名古屋の来場者全員に配布しました。デザインのモチーフは当初ソラちゃんのデビューシングルをイメージし、デビュー曲のタイトルを私が勝手に「機上の乙女の電子工作」としたのですが、いつの間にか没案

3　教育テレビの「R の法則」や舞台などにご出演され、現在はアイドル活動をされています。

図 3-8: OSC 名古屋と京都で配布したうちわ

となりデビューシングルの面影だけ残したうちわ（図 3-8）となりました。

OSC 名古屋出展のあと、じむさんから「伊藤さんが FlashAir に GPIO を搭載させた」と聞きました。「穴を開けられなかったのでは？」と聞くと、「開けてませんよ」と。まさか穴を開けずに SD の端子を GPIO 化してしまうとは、想像もしませんでした（汗）。GPIO として利用する際は、FlashAir の中のフラッシュメモリにホスト機器側からデータを読み書きすることができません（Wi-Fi からは可能）が、GPIO が搭載されたことには変わりなく、まさに Dr. K 氏にその必要性を指摘されてから 1 年越しの実現でした。伊藤さんが 1 月の Edison の話を意に介さなかったのは、既にこれを仕込んでいたからかもしれません。

ソラちゃん基板の作成

8 月の OSC 関西@京都ではこの GPIO 機能を宣伝することになり、じむさんが活用事例として FlashAir を使ったワイヤレスなクレーンゲームを作りました（図 3-9）。これが大好評でした。

図 3-9: ワイヤレスクレーンゲーム

この事例製作の過程でじむさんは「ユーザーに FlashAir の GPIO 機能を気軽に利用してもらうためには、FlashAir とそれに接続したい機器とを簡単に接続できるインターフェースを用意したほうがよい」ことに気づきました。確かに、この世の中に FlashAir の GPIO 機能に対応した SD カードスロットなど存在しません。じむさんは、組み込み用途を想定して Arduino のシールド としても使える基板を設計しました（図 3-10）。そして次の 10 月の OSC 東京（秋）において「遊んでみる気のある人に基板を使って頂き、twitter などで活用事例を報告してもらおう」と考えました。ちなみにじむさんが基板にソラちゃんのイラストを入れたのは、同人サークル「れすぽん」の余熱さんがプリント基板の配線で巧妙に女の子のイラストを描いた同人基板を設計され（図 3-11）、これに感化されたからでした。

さて、基板の配布について当初から「作るのに失敗した人がクレームを出してくるのではないか?」という懸念がありました。土壇場でリスク回避のために配布をやめようか、という話になりかけましたが、せっかく設計した基板を作ってもらえないのはもったいない限り。そこでじむさんに「余熱さんに相談してみてはどうか?」と提案してみました。

2日くらい経ったころでしょうか。じむさんから突如、衝撃のメールが来ました。「余熱さん、うちの社員でした!」と。結局10月のOSC東京にて、諸注意を守り、必ず作成頂き、twitterなどでつぶやいて頂くことを条件に基板を試験配布しました。作成レポートが次々とtwitterにてつぶやかれ、ほとんどの方がLEDをチカチカ点灯させる「Lチカ」に取り組んでくださいました。

図 3-10: GPIO のための
インターフェイス基板製作例

図 3-11: 余熱さん作の「基板少女」

その後、余熱さんもGPIO利用のためのインターフェース基板を独自に設計されました。余熱さんのお友達の村口さんも、外部からFlashAirに転送した画像を液晶に表示させる装置を作り、人間の音読よりも低速な転送速度1.8bpsで液晶画面にソラちゃんを描画させることに成功し、新たな応用例を示してくださいました。このお二人の作成記事は本誌にてお読みいただけます。

おわりに

そんなこんなで迎えたこのMaker Faire Tokyoで配布された本誌ですが、同人誌製作の経験がある余熱さんのご尽力で完成しました。OSCを通したFlashAirチームの活動の顛末をつづってみましたが、いかがでしたか?　FlashAirはユーザーの皆様の声でこれからも進化する余地があります。もし我々の活動に興味をお持ち頂けるようでしたら、twitterのソラちゃんアカウント(@Hirameki_Sora)をフォロー頂きまして、ご意見をつぶやいて下さいますと大変幸いです。

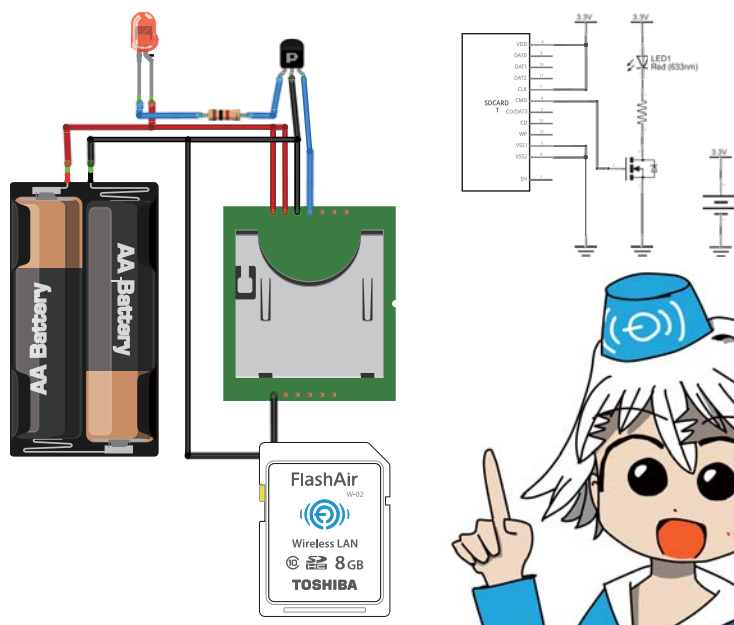
FlashAir 評価基板 Airio(えありお) の設計

余熱 @ れすぽん

皆さまこんにちは、余熱です。普段はサークルれすぽんという同人ハードウェアサークルで基板の設計などしています。今回は、FlashAir の GPIO 機能を簡単に試すことができ、Arduino とも簡単に接続可能な基板 Airio(えありお) を製作しましたのでそれについて書こうと思います。設計についての細かい話ではなく、製作記として軽い感じで書いていくつもりです。なお、基板の取扱説明書、回路図、ソフトウェアなどは web ページに置いてありますので適宜参照してください (http://yone2.net/)。

FlashAir の GPIO 機能

FlashAir は本来、マイコンなどとセットで使用しますが、隠しフォルダ内の設定ファイルを書き換えることで SD インターフェース端子を I/O として利用することができます。この機能により FlashAir の端子をスマホなどから動かすことが可能です。LED をチカチカさせるための回路も下記のように非常に単純になります。

この機能は Class10 の FlashAir(FW Ver. 2.00.03 以上) で利用可能です。詳細は FlashAir Developers(https://flashair-developers.com) を参照してください。

※画像は fritzing(http://fritzing.org) によって生成しました。

企画

FlashAir の GPIO 機能を簡単に試すために LED チカチカ (L チカ) の基板が必要だと思い、設計を開始しました。小ロットで製作してイベントで頒布するのが最終的な目標です。設計開始直後のメモには「低価格に振ったもの : FlashAir + LED/ スイッチ + 電源 だけ。」と、ありました。その後、FlashAir と LED を単に繋いだだけでは面白くないため、Arduino シールドとしての機能を持たせることにしました。設計当初のラフスケッチを図 4-1 に貼っておきます。

図 4-1: 設計当初のラフスケッチ

イベントでの頒布を視野に入れていたため、なるべく低コストを目指しました。電源は USB 給電とし、基板を USB コネクタに直挿しできるようにしてあります。基板は海外の基板製造工場の最小サイズである 5 × 5cm に収まるように設計します。5cm 四方の基板ではギリギリ Arduino シールドのサイズには足りないのですが、ICSP コネクタを利用して接続しようと考えました (図 4-2)。

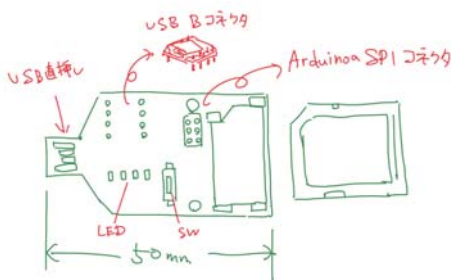
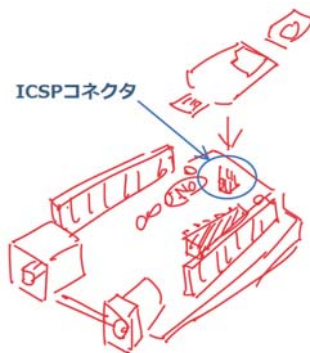



図 4-2: Arduino と ICSP コネクタで接続

設計

部品の選定から回路設計、レイアウトまではおよそ 2 週間程度かかりました。イベント頒布の場合は手実装することを考慮し、0.5mm 以下の狭ピッチ部品は使用せず、入手性の高い部品を選定しました。回路設計、レイアウトは KiCad[1] を使って行いました。

電源周り

最初に電源周りをまとめておきます。基板の入力電圧は USB 給電もしくは Arduino からの 5V 給電です。また、Arduino の信号レベルは 5V です。一方、SD カードは 3.3V 駆動ですので、電源にはレギュレータが、信号にはレベル変換が必要になります。レギュレータは、JRC の NJU7223DL1 を選定しました。出力電流は 500mA です。信号レベル変換については専用の IC を使うこともできるのですが、簡易的に抵抗分圧にてレベル変換を行っています。

1　http://kicad.jp/

LED を駆動させる回路

FlashAir から FET を介して LED を駆動します。フルカラー LED は順電圧が高いため、駆動電圧は 5V にしました。FET を並列に配置することで、FlashAir からでも Arduino からでも LED を駆動できるようにしました (図 4-3)。LED の電流制限抵抗については、机上の計算の後にブレッドボード上で明るさを確認し、値を確定させました。

図 4-3: LED 駆動部の回路図

SD カードスロットの部品選定

SD カードスロットの選定については、かなりの時間と手間かかりました。要求としては、「基板面積を考慮し、なるべく短く、かつ、実装性を考慮し、金属部分がなるべく少ないもの」だったのですが、取扱店がそもそも少なく、Digi-key などでは高価、秋葉原で買える部品は在庫数量に不安がある、という状況でした。そんな時に見つけたのが Raspberry Pi 用の SD カードスロット[2]です (図 4-4)。日本ではレオコム[3]から購入することができます。

図 4-4: Raspberry Pi と同じ SD スロット

FlashAir のピンアサイン

SD カードは、デジカメなどの機器との通信では SD カード I/F を使用しますが、Arduino などのマイコンと接続する場合には SPI を使用します。FlashAir の GPIO 機能を使う場合にも、使えるピンが決まっていて、SPI で使用する端子とは別の端子が割り当てられています。表 4-1 にまとめておきます。

Airio では、SPI と GPIO の両方のモードをサポートします。

表 4-1: FlashAir のピンアサイン

ピン	SD I/F	SPI	GPIO
8	DAT1		0x04
7	DAT0	DO	0x02
6	Vss2		
5	CLK	SCLK	
4	Vcc		
3	Vss1		
2	CMD	DI	0x01
1	DAT3	CS	0x10
9	DAT2		0x08

2 http://uk.farnell.com/multicomp/412d02f-09pc003sv/receptacle-sd-card-smt/dp/2226409
3 http://www.leocom.jp/

回路図とレイアウト

回路が決まったらレイアウト作業に入ります。この段階で、基板を USB 直刺しにすると 5 × 5cm に収まらないことが発覚し、USB-A コネクタをつけることになりました。最終的なレイアウトを図 4-5 に示します。

基板は両面基板で配線し、FusionPCB にて製造しました。部品実装はトースターを改造した自宅のリフロー炉にて行いました。イベントで頒布するような小ロットであればこの程度で十分対応可能です。

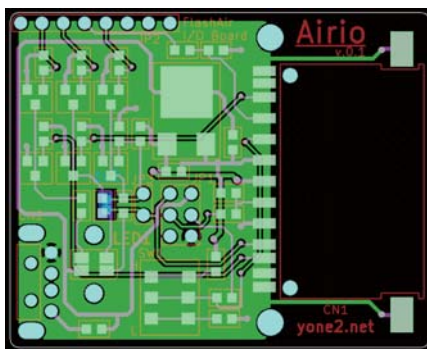



図 4-5: Airio のレイアウト

動作確認

GPIO 機能の確認

GPIO 機能を試すためには、FlashAir に HTML ファイルを書き込み、スマホなどからアクセスします。フルカラー LED は 7 色出すことができるため、それぞれの色のボタンを配置し、タップした色が光るようにしました (図 4-6)。

Arduino との接続

Arduino と接続する際には、ピンヘッダを取り付け、裏面の半田ジャンパをパターンカットする必要があります。サンプルコードを Arduino に書き込めば、SD カードにアクセスすることが可能です。また、フルカラー LED は Arduino 接続時には Arduino 側から PWM 駆動することが可能です。

図 4-6: GPIO 機能の動作確認

おわりに

設計開始から完了 (ガーバ out) まで、2 週間程度しかなかったため、かなり突貫で作ったのですが、届いた基板が無事動作した時は安心しました。ハードを作る場合には、回路設計そのものよりも部品選定・調達に気を使うことが多く、そのあたりのノウハウもいずれまとめたいと思っています。

Arduino で FlashAir を制御せよ！

土居

はじめに

FlashAir は、無線 LAN アダプタと Web サーバーが一体化したスーパー SD メモリーカードです。Web サーバーを内蔵しており、保存したファイルを無線 LAN 経由で取り出したり、特別な URL にリクエストを送れば端子を GPIO として使えたりしてしまいます。

しかし、FlashAir の応用力はそれだけではありません。実は、無線 LAN 機能を自由自在に制御する方法があるのです。

それが、SD インターフェースの拡張コマンドである「iSDIO」です。これをつかえば、マイコンボードなどの SD ホスト機器から、FlashAir の無線 LAN 機能の高度で精密な制御が行えます。FlashAir が HTTP 通信コプロセッサになってしまうのです。

FlashAir Developers (https://flashair-developers.com) では、Arduino に、iSDIO クライアント機能を実装して FlashAir を制御し、近傍の無線 LAN AP のスキャンと接続から、Web サイトのデータをダウンロードするまでの方法を、チュートリアル形式で紹介しています。 本記事はそのダイジェスト版です。 いったいどんなことができるのか、その可能性の一端を感じていただければと思います。

iSDIO

iSDIO (Intelligent SDIO) とは、SD メモリーカードの規格団体である SD アソシエーションによって定められた、FlashAir のような拡張機能付き SD メモリーカードをコントロールするための新しい規格です。

通常、Arduino のようなマイコン機器に無線 LAN 機能を付ける場合、無線 LAN アダプタ部品とドライバソフトウェアが必要になります。しかし、iSDIO をつかえば、マイコン機器に代わって FlashAir に無線 LAN 通信を代行させることができます。FlashAir が無線 LAN アダプタとドライバソフトウェアが一体化した部品になると考えてもよいかもしれません。SD 規格は、会員企業以外にも簡易版仕様書 (Simplifed Specifications - SD Association) が公開されており、iSDIO の仕様はこのなかの Part E7 Intelligent SDIO Simplified Specification に定義されています。

実際には、iSDIO 仕様とはコマンドやりとりの手順 (プロトコル) のみを定めた枠組みのようなものであり、利用できる機能や引数などの詳細は拡張機能の種類(アプリケーション）ごとに Addendum（補遺）で定められます。 FlashAir は無線 LAN 内蔵カード向けの「Wireless LAN Addendum」に準拠しています。

iSDIO レジスタのメモリ配置
(SD Specifications Part E7 iSDIO Simplifed Specification Version 1.10 より抜粋)

Address	Name	Short Description	Type
00000h	Command Write Register Port	Data Port to write the iSDIO Command Write Data	W/O
00001h - 001FFh	Reserved		
00200h	Response Data Register Port	Data Port to read the iSDIO Command Response Data	R/O
00201h - 03FFh	Reserved		
00400h - 005FFh	Status Register	Memory Area for iSDIO Status Register	Table 2-7
00600h - 007FFh	Capability Register	Memory Area for iSDIO Capability Register	R/O
00800h - 00FFFh	Reserved		
01000h - 01FFFh	Reserved for Vendor		
02000h - 1FFFFh	Reserved		

Table 2-6 : iSDIO Register Map

iSDIO のコマンドは、iSDIO レジスタの読み書きによって行われます。

000h ～ 1FFh 番地への書き込みがコマンド発行になります。また、コマンドレスポンスは 200h ～ 3FFh の領域に返されます。400h ～ 4FFh は iSDIO 共通ステータスです。コマンドの実行状況などが取得できます。500h ～ 5FFh 番地は Addendum で定められるアプリケーションごとのステータスを定義するエリアとなっており、FlashAir ならば無線 LAN 機能の各種情報が取得できます。

ステータスを読み取る

まず FlashAir のステータスを読みとってみたいと思います。 400h 番地からの領域を読み取り、ステータスマップに従ってよみやすく表示します。

```
card.readExtMemory(1, 1, 0x400, 0x200, buffer);
... 個々のステータスを読みやすく表示するコードが続く ...
```

これを実行するとたとえば下記のような出力が得られます。

```
== iSDIO Status Registers ==
[0400h] Command Write Status:
[0420h] iSDIO Status: CRU
[0422h] iSDIO Int Enable:
[0424h] Error Status:
[0426h] Memory Status: MEX
[0440h] Command Response Status #1: id = 3, sequence id = 0, status = Process
Succeeded
... (snip) ...
```

AP モードでの起動

AP モードで無線 LAN 機能を ON してみましょう。 Establish コマンド (ID 03h) を使用します。 同時に、HTTP サーバー機能と DHCP サーバー機能も立ち上がります。

```
uint8_t* p = buffer;
p = put_command_header(p, 1, 0);
p = put_command_info_header(p, 0x03, sequenceId, 3);
p = put_str_arg(p, "sdiotest");
p = put_str_arg(p, "12345678");
p = put_u8_arg(p, 0x06);
put_command_header(buffer, 1, (p - buffer));
card.writeExtDataPort(1, 1, 0x000, buffer);
```

2 行目 ヘルパー関数を利用して、コマンドのヘッダを作っています。 最後の引数にはコマンドデータの長さ (バイト数) を入れますが、可変長の引数があるため、後で計算して上書きします。

3 行目 ヘルパー関数を利用して、コマンド情報のヘッダを作っています。 Establish のコマンド ID0x03 、シーケンス IDsequenceID 、引数の個数 3 を指定しています。

4 行目 1 番目の引数「SSID」を書き込んでいます。 文字列引数を書き込むヘルパー関数を利用しています。

5 行目 同様に、2 番目の引数「ネットワークキー」を書き込んでいます。

6 行目 3 番目の引数「セキュリティモード」を書き込んでいます。 ここでは、WPA2-PSK and AES を表す 0x06 を指定しています。

7 行目 全データが書き込まれ、バイト数が確定したので、コマンドヘッダを改めて書き込んでいます。

8 行目 FlashAir に対してコマンドデータを書き込んでいます。書き込みが完了すると、処理が始まります。

実行してステータスを表示してみましょう。 WLAN ステータスが AP および IP Address が 192.168.0.1 となっていることから、FlashAir が AP モードで立ち上がっていることが確認できます。

```
[0440h] Command Response Status #1: id = 3, sequence id = 3, status = Process
Succeeded
... (snip) ...
[0506h] WLAN: No Scan, No WPS, Group Client, AP, Infrastructure, No Connection,
[0508h] SSID: sdiotest
[0528h] Encryption Mode: WPA2-PSK and AES
[0529h] Signal Strength: 0
[052Ah] Channel: 11
[0530h] MAC Address: E8E0B758A7FB
[0540h] ID:
[0550h] IP Address: 192.168.0.1
[0554h] Subnet Mask: 255.255.255.0
[0558h] Default Gateway: 192.168.0.1
[055Ch] Preferred DNS Server: 192.168.0.1
[0560h] Alternate DNS Server: 0.0.0.0
```

無線 LAN のスキャン

Scan コマンド (ID 01h) で近傍にある無線 LAN AP の SSID を検索してみましょう。 実行すると近くにある無線 LAN の SSID と信号強度、暗号化方式がわかります。

```
uint8_t* p = buffer;
p = put_command_header(p, 1, 0);
p = put_command_info_header(p, 0x01, sequenceId, 0);
put_command_header(buffer, 1, (p - buffer));
card.writeExtDataPort(1, 1, 0x000, buffer);
```

スキャン結果は、コマンド完了後に結果レジスタ (200h 番地) を読み取ることで取得できます。

```
Number of APs: 5
mynetwork, B86B23663750, 70, WPA2
flashair_led, B86B23583750, 64, NoSec
flashair203r, B86B23005049, 61, WPA2
HWD14_VEGETA, C40528C98B0C, 52, WPA2
FlashairT5-v2, E8E0B744A7FB, 49, WPA2
```

Web ページの取得

https://flashair-developers.com/ から HTML データを受信してみましょう。 HTTP 通信コマンドには、プロトコルやデータの渡し方の違いなどにより、いくつかのバージョンがありますが、今回は HTTP over SSL (Secure Socket Layer) を使用するため、SendHTTPSSLMessageByRegister コマンド (ID 23h) を使います。

```
uint8_t* p = buffer;
p = put_command_header(p, 1, 0);
p = put_command_info_header(p, 0x23, sequenceId, 2);
p = put_str_arg(p, "flashair-developers.com");
p = put_str_arg(p,
"GET /en/ HTTP/1.1¥r¥n"
"Host: flashair-developers.com¥r¥n"
"User-Agent: Mozilla/5.0 (Windows NT 6.3; WOW64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/36.0.1985.125 Safari/537.36¥r¥n"
"¥r¥n");
put_command_header(buffer, 1, (p - buffer));
card.writeExtDataPort(1, 1, 0x000, buffer);
```

4 行目 サーバーのドメイン名を指定します。IP アドレスも指定できます。
5 行目～ HTTP のリクエストヘッダを生成しています。

レスポンスの受信

コマンド完了を待機してから、取得したデータを読み取ります。 事前にインターネット通信可能な無線 LAN に接続しておいてください。200h ～ 3FFh 番地に書かれているサーバーから返されたデータを表示してみましょう。

```
HTTP/1.1 200 OK
Date: Tue, 21 Oct 2014 06:25:10 GMT
Server: Apache/2.2.15 (CentOS)
... (snip) ...
<title>FlashAir Developers - Home</title>
... (snip) ...
```

詳しくは FlashAir Developers へ

超駆け足ですが、iSDIO で FlashAir を制御して、無線 LAN 機能を Arduino から使う方法を紹介してきました。 その他、https://flashair-developers.com/ では、無線 LAN の切断、ステーションモードでの接続や、ヘルパー関数のサンプルコードなどを掲載しています。ぜひご覧ください。

FlashAir で無線 SPI Master

村口

はじめに

本記事では「FlashAir が無線 SPI Master としても使えること」をご紹介します。まず、SPI の概要説明を行います。続いて、FlashAir と、SPI 液晶モジュールの接続例を挙げ、ソフトウェアによる制御例をご紹介します。

Aruduino などから FlashAir を制御する際には、FlashAir は SPI Slave として機能しますが、本記事では、FlashAir を SPI Master として利用するお話になります。ご留意ください。

さあ、FlashAir で、電子工作の幅を広げていきましょう！

SPI について

SPI (Serial Peripheral Interface) は、Motorola 社 (現 Freescale Semiconductor) が提唱した周辺デバイス向けの全二重シリアル通信バス規格です。SPI バスは、SCLK, MOSI, MISO, SS の 4 種類の信号線で構成されます。表 6-1 に各信号線の説明を記載しました。

表 6-1: SPI の信号線

信号名	送信側	受信側	説明
SCLK (Serial Clock)	Master	全 Slave	SCLK に同期してデータ転送が行われます。SCLK のエッジやデータとの位相関係は、後述する SPI モードで 4 種類のバリエーションが存在します。
MOSI (Master Out Slave In)	Master	全 Slave	Master から Slave にシリアルデータを送信します。
MISO (Master In Slave Out)	各 Slave	Master	Slave Select によって選択された Slave から Master にシリアルデータを送信します。非選択状態の Slave は、他の Slave の通信とぶつからないように、MISO を Hi-Z にする必要があります。
SS (Slave Select)	Master (通常は Slave 毎に独立した SS が必要)	各 Slave	負論理です。L で対象 Slave 選択状態を示し、H で対象 Slave の非選択状態を示します。SS は、Slave デバイスの数だけ必要になります。

SPI Master 側でマスタークロックのタイミングを制御可能なので、GPIO のソフトウェア制御でも SPI Master として動作させることができます。実際の SPI Slave デバイスでは、SPI で定義されている信号以外にも、リセット端子など、追加で制御を必要とする端子が存在する場合があります。

SPI バス構成

SPI バスに配置できる Master 数は 1 個のみです。マルチバスマスタ構成にはできません。SPI バスに配置できる Slave 数には制限がありませんが、Slave デバイス毎に独立した SS 線が必要になります。SCLK, MOSI, MISO は、全 Slave で共有されますが、SS の信号本数は Slave デバイス数に比例するため、Slave デバイス数が多いシステムでは、使用する信号本数が大きくなる欠点があります。図 6-1 に、一般的な SPI バスの構成例を示します。

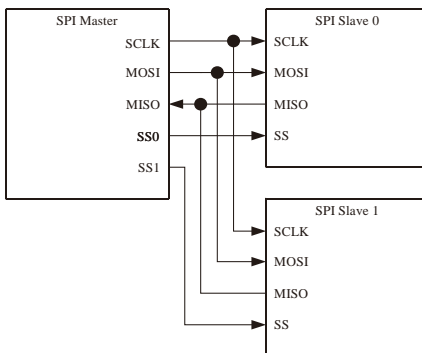



図 6-1: SPI バス構成例

SPI 動作モード

SPI 動作モードには、クロック極性や位相関係の違いから、4 つの動作モードがあります。SPI モードと、クロック極性 CPOL、クロックフェーズ CPHA の関係を表 6-2 に示します。

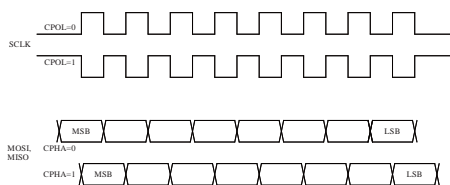
表 6-2: SPI モードと、クロック極性 / クロック位相の関係

SPI モード	クロック極性 CPOL 設定	クロックフェーズ CPHA 設定	説明
0	0	0	SCLK の L 期間中に MSB データが送信される。 SCLK の最初の立ち上がりに同期してラッチされる。 SCLK の立ち下がりに同期して次のデータが送信される ...
1	0	1	SCLK の最初の立ち上がりに同期して MSB データが送信される。 SCLK の立ち下がりに同期してデータがラッチされる。 SCLK の立ち上がりに同期してデータが送信される ...
2	1	0	SCLK の H 期間中に MSB データが送信される。 SCLK の最初の立ち下がりに同期してラッチされる。 SCLK の立ち上がりに同期してデータが送信される ...
3	1	1	SCLK の最初の立ち下がりに同期して MSB データが送信される。 SCLK の立ち上がりに同期してデータがラッチされる ...

クロック極性 CPOL は、データ転送開始時のクロックトグルが、立ち上がりから始まる (CPOL=0) か、立ち下がりから始まる (CPOL=1) かを示しています。

クロックフェーズCPHAは、クロックとデータの位相関係を示しており、CPHA=1設定は、CPHA=0 設定のデータ位相を、後ろに 180 度遅延させたような位相関係になっています。

図 6-2: SPI モードのタイミング波形

CPOL, CPHA の関係のタイミング波形を図 6-2 に示します。

FlashAir と LCD モジュールの接続

今回は、2014 年 11 月現在、Aitendo さんで入手可能な「STN 液晶モジュール (100x32) HQ077BC」を使用しました。この液晶モジュールは、コントローラに KS0713 を使用しています。3.3V 単一電源で駆動でき、SPI で制御可能です。SPI の仕様は、ワードサイズ 8bit、SPI モード 0 です。この液晶モジュールは SCLK 端子 , MOSI 端子 , SS 端子に加えて、リセット端子 RESETN、レジスタセレクト端子 RS の計 5 本の IO 制御が必要になります。MISO 端子は存在しません。FlashAir も 3.3V で動作し、GPIO も 5 本あるので、そのまま接続可能です。LCD モジュールと FlashAir の接続は、表 6-3 のように行いました。

この液晶モジュールの端子は、0.5mm ピッチ、19 ピンのフレキなので、2.54mm ピッチに変換するブレークアウト基板を利用しています。また FlashAir 側の CLK 端子[1]については、プルアップ、プルダウン処理を実施し、入力の値を固定しています。

表 6-3: LCD モジュールの接続先

LCD モジュール ピン		接続先
1	VSS	GND
2	VDD	3.3V
3	CS1B(SPI SS)	FlashAir Pin2, CMD
4	CS2(L or H 固定)	H 固定
5	RS(0: command, 1: data)	FlashAir Pin7, D0
6	RD(L or H 固定)	H 固定
7	WR(L or H 固定)	L 固定
8	PS(0: serial, 1: parallel)	L 固定
9	RESETN	FlashAir Pin8, D1
10	MI(0: 8080, 1: 6800)	L 固定
11	DB7(SPI MOSI)	FlashAir Pin9, D2
12	DB6(SPI SCLK)	FlashAir Pin1, D3
13 ～ 18	DB5 ～ DB0	NC
19		NC

1 FlashAir を用いたハードウェア設計のガイドラインを、今後 FlashAir Developers に掲載する予定です。 https://flashair-developsers.com

GPIO 制御

FlashAir に無線 LAN 接続した PC の Ruby スクリプトから HTTP リクエストを発行して FlashAir の GPIO 制御しました。GPIO の信号を変化させる度に HTTP クエリ文字列を再生成して、HTTP リクエストを行うようにします。Ruby の open は、HTTP も扱うことができ、処理が完了するまでブロックするので、途中で実行順序が入れ替わったりすることはありません。下記に、Ruby の open で HTTP リクエストを発行するコード記述例を示します。

```
コード記述例：
require 'open-uri'
…
open(sprintf("http://flashair/command.cgi?op=190&CTRL=0x1f&DATA=0x%02x",
        @clock<<4 | @data<<3 | @reset<<2 | @rs<<1 | @csb))
...
```

続いてリセット解除シーケンス、レジスタライト関数、データ転送関数を作成し、LCD モジュールの初期化シーケンスを作成します。

今回使用した液晶モジュールは、液晶全体がページと呼ばれる領域で 4 分割されており、1 ページは、100x8 ドットで構成されています。SPI の 1 ワードのライトアクセスで、ページ内の 1x8 ドット単位の書き換えが可能です。ページ内のアドレスは自動インクリメントされるため、100 ワードのライトアクセスで 1 ページ分の描画が可能です。100x32 ドットの画面全体を書き換えるには、液晶モジュールの初期化シーケンスと、400 ワードのライトアクセス、ページアドレス指定が 4 回が必要になります。

FlashAir で閃ソラを表示

まず、100x32 ドットのモノクロ画像を用意しました。その後、ImageMagick や Cairo ライブラリなどを利用して液晶モジュールの最小描画単位である 1x8 ドットごとのピクセル情報を取り出して、順番に描画すると図 6-3 のようになります。

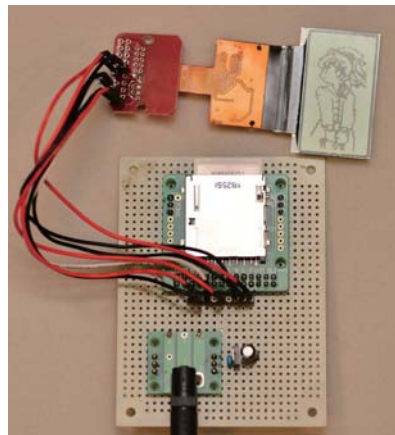



図 6-3: FlashAir で閃ソラを表示

FlashAir で Hello, World!

Hello, world! を行うにあたり、まず 5x5 ドットの極小 ASCII フォントを作成しました。文字と文字の余白を 1 ドットとし、6x6 ドットに 1 文字を収めるようにしています。このフォントを使用すると、100x32 ドットの画面内に、16 文字 x 5 行で、最大 80 文字を描画することができます。配列にフォントデータを格納する際は、ASCII コードの順番通りに収めておくと、フォントデータを利用するとき便利です。実行結果は図 6-4 のようになりました。

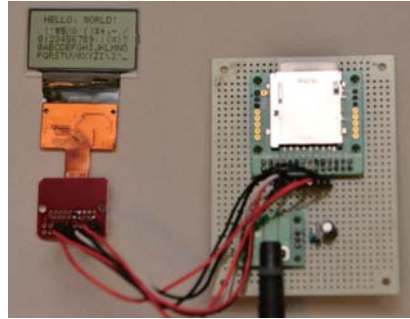



図 6-4: FlashAir で Hello, world!

FlashAir の無線 GPIO による SPI 転送レート

FlashAir の無線 GPIO を利用したソフトウェア SPI では、100x32 ドットの全面を書き換えるのに、IO トグル (HTTP リクエスト) が 7724 回必要となり、時間にしておよそ 35 分を要しました。FlashAir の無線 GPIO を利用した SPI 転送レートは、約 1.8bps ですが、低速の M2M 用途では問題なく利用することができるでしょう。

ガラガラ
FlashAir
はっはっ
ガラガラ
ピロピロ
んっもー
はい

キョロキョロ
そ〜〜
閃さん、早く工作員モードになってくださいっ。でないとミッション…
やっとるわい
こっちもね、一生懸命着替え場所を探してるんですよ！
パ
シュ
FlashAir
ピロ〜ン
ミッション
Raspberry PiでFlashAirを動かす解説をせよ。
ほ〜このミッション、引き受けましょう。
電子工作員
参上
じも

ネットからRaspbianのイメージを
ダウンロードしてきます。

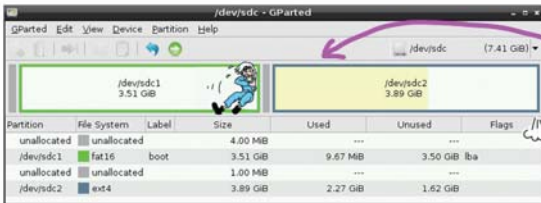
ダウンロードしたイメージファイルは
ddコマンドでUSBカードリーダライタの
FlashAirへ展開します。

（例）

```
$sudo dd if=/home/pi/Desktop/flashair/
        2014-09-09-wheezy-rasobian.img of=/dev/sdc
```

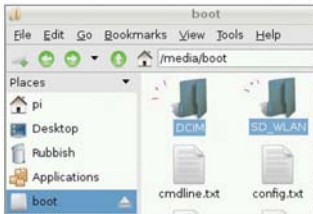
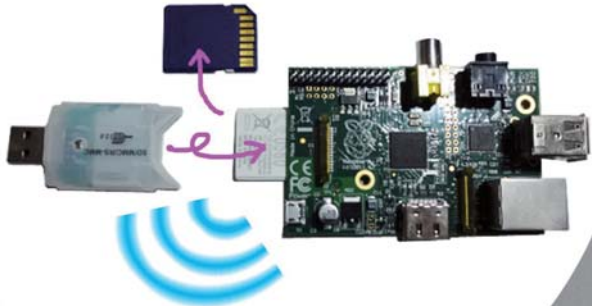
東芝サイトにある"PC設定ツール"で
初期化したばかりのFlashAirを
USBリーダライタに挿して、2つの
フォルダ("¥DCIM","¥SD_WLAN")を
ブートSDカード側へ
コピーしておきます。

特に"¥SD_WLAN"フォルダは
隠しフォルダになっていますので
気をつけましょう。

パーティション編集アプリ(=gparted)を使って、
FlashAirの各パーティション領域(sdc1/sdc2)を
マウントを外して変更します。

sdc2のext4領域を後ろに下げて、空いた部分を
sdc1のfat16領域を広げる(※)感じです。

Raspbianのイメージの展開で、
最初のパーティション構成が
このようになっていました。

前半の「fat16」(黄緑色)と、
後半の「ext4」(青碧色)です。

広くなったsdc1(fat16/32側)を再度マウントして、そこに最初に退避しておいた
2フォルダ("¥DCIM","¥SD_WLAN")をコピーします。

※fat16領域は容量=4G以上設定すると、fat32にフォーマットさせられます。
このとき"/dev/sdc1"のファイルも消えちゃうので、そうしたい場合は事前に全データを
一旦退避させて、フォーマット後コピーして元に戻します。

■ FlashAir Doujinshi - FlashAir の同人誌

2014 年 11 月 23 日 初版第 1 刷発行
2014 年 12 月 30 日 初版第 2 刷発行
著者 : 高田 / 伊藤 /Pochio/ 余熱 / 土居 / 村口 / じむ
表紙イラスト : じむ
表紙デザイン : 余熱 / じむ
編集 : 余熱
発行 : FlashAir Developers
連絡先 : support@flashair-developers.com

あとがき

寄稿はあらためて活動の意義を考える良い機会になりました。活動のきっかけになった MTG を「秋葉原のおたくの方々との打合せ」と会議招集したことも懐かしい思い出です。

高田

FlashAir でもっと色々と出来るようにしたいですねえ。

伊藤

この拡販活動で初めてタレント事務所に電話したり、声優さんにインタビューされたりと貴重な体験をしました。今後は組み込み用途で多用されてほしいです。制作事例はぜひ #FlashAir を付けてツイッターに！

Pochio

一生懸命書いたので、楽しんでもらえると嬉しいです。

余熱 @ れすぽん

FlashAir コミュニティの構築を目指して始めた FlashAir Developers ですが、とうとう同人誌ができるまでになりました。感動＆感謝です。サイトの Arduino チュートリアルも読んでね！

土居

FlashAir は、無線 SPI Master としても利用できます。この可能性の広がりにワクワクして頂ければ幸いです！

村口

Microsoft や GMO と違うのは、キャラの内製化率（※）が 100%。T 社らしいですね。
（※自社の製品を構成する部品のうち、外部に委託・発注せず、自社が製造・制作した部品が占める割合。）

じむ

初めはたった数人でスタートした FlashAir ですが、いろんな場所で、仲間が増えてきたことを実感しています。もっともっとファンが増えるようなプロダクトに育てます！

笠原